Software Requirements Specification

FinXpert

Version <1.0>

Prepared by

Group 16:		**Group Name: Tech Titans**
Aniket Kumar Choudhary	**220140**	**aniketkc22@iitk.ac.in**
Archit Atrey	**220195**	**aarchit22@iitk.ac.in**
Ayush Kumar Yadav	**220267**	**ayushku22@iitk.ac.in**
Devank Saran Prajapati	**220339**	**devanks22@iitk.ac.in**
Dwij Om Oshoin	**220386**	**dwij22@iitk.ac.in**
Karthik S Pillai	**220502**	**karthiksp22@iitk.ac.in**
Madhav Khetan	**220596**	**madhavk22@iitk.ac.in**
Piyush Meena	**220768**	**piyushm22@iitk.ac.in**
Rahul Mahato	**220859**	**mrahul22@iitk.ac.in**
Sumit Kumar	**221102**	**sumitkumar22@iitk.ac.in**

Course: **CS253**

Mentor TA: **Naman Baranwal**

Date: **24/01/2025**

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Tech Titans	Initial Draft	24/01/25

1 Introduction

1.1 Product Scope

The platform aims to revolutionize how users manage their finances by providing traditional tracking of expenses and incomes as well as deep insights and tips into their financial behavior. The goal is to empower users with knowledge that helps them make informed financial decisions, understand their financial patterns and in turn improve their financial stability.

Key Components and Features

Spending Habits Analysis:

Expense Categorization: Automatically categorize expenses based on user input into various predefined and user-defined categories (e.g., groceries, entertainment, utilities).

Trend Analysis: Identify and visualize spending trends over time, highlighting areas where spending is varying and how much it has increased or decreased compared to a previous value either user defined or set by the software using values from a previously recorded time. The platform also aims to identify seasonal and yearly variations in spending.

Financial Behavior Insights:

Behavioral Patterns: Analyze users' spending behavior to identify patterns, such as frequent impulse purchases, seasonal spending spikes, and habitual expenditures.

Financial Goals: Help users set and track financial goals, such as paying off bills and subscriptions and expenses like loans, insurances etc., planning a vacation, or building an emergency fund. The platform can provide actionable recommendations to achieve these goals categorizing them depending on a user defined list or according to their importance (e.g. paying off loans and debts will have a higher priority than vacation saving).

Track payments across multiple bank accounts: The platform aims to be capable of tracking payments across multiple accounts a user might own. This includes updates on pending payments on credit cards linked to different bank accounts, information on expiring gift points on the user's gift cards etc.

Psychology of Spending:

Mood Tracking: Allow users to log their moods and emotions, correlating them with their spending habits to uncover emotional spending patterns.

Comprehensive Financial Tracking:

Real-Time Updates: Ensure that financial data is updated in real-time, providing users with an accurate and current view of their financial situation.

Interactive Dashboard: An interactive dashboard that displays financial data visually, making it easier for users to understand their financial health at a glance.

Smart AI/NLP-Powered Chat Assistant:

Natural Language Processing (NLP): Enable users to interact with the platform using natural language, making it more user-friendly and accessible.

Conversational Interface: Provide a chat-based interface for users to add expenses instead of using the interface directly, request financial summaries, and receive personalized insights. The chatbot can help the user to take informed financial decisions by providing updates on pending payments, information on expiring credit or gift card points etc.
AI-Driven Summaries: Generate AI-driven financial summaries that analyze users' spending habits and offer tailored advice.

Secure and Efficient Authentication:

OTP Authentication: Implement rate-limited OTP authentication to ensure security.

Account Management: Provide smooth recovery mechanisms and notifications for account lockouts due to exceeded OTP requests.

Data Privacy: Ensure that all user data is securely stored and encrypted.

1.2 Intended Audience and Document Overview

This document is crafted for various stakeholders involved in the Finance Management App, catering to the diverse needs of users, developers, project managers, marketing staff, testers, and documentation writers.

Users
Explore how to interact with the app for financial management, including creating accounts, adding entries, and viewing dashboards.

Developers
Find detailed descriptions of key functionalities, security measures (such as OTP authentication and data privacy), and Smart AI Assistant interactions to guide the app's development.

Project Managers
Understand the project scope, goals, and timelines. Gain insights into app features and benefits for effective monitoring and resource allocation.

Marketing Staff
Comprehend the app's unique selling points, such as the Smart AI Assistant and secure financial data analysis, to create promotional content and campaigns.

Testers
Review functionalities like OTP authentication, guest login, and interactive features for thorough testing of usability and security.

Documentation Writers
Refer to the guide for creating manuals, tutorials, and FAQs, focusing on intuitive app use for different audiences.

Document Overview:
Introduction:
Provides an overview of the Finance Management App, its purpose, and benefits.
Product Scope:

Describes the scope of the project, highlighting the objectives and goals.

Intended Audience and Document Overview:
Identifies the target readers and outlines the structure of the document.

System Architecture:
Offers an in-depth view of the system's architecture, including the database design and key components.

Functional Requirements:
Breaks down the system's functionalities based on user roles, detailing the expected behavior of each feature.

Non-Functional Requirements:
Discusses performance, security, and usability requirements that go beyond specific functionalities.

User Interface Design:
Presents the user interface design, including wireframes and design principles.

Testing Requirements:
Outlines the testing strategy, including test cases, scenarios, and expected outcomes.

Deployment and Maintenance:
Provides guidelines for deploying the system and ongoing maintenance considerations.

Appendix:
Includes supplementary information, such as glossary, acronyms, and any additional documentation references.

Sequence for Reading:
Readers are recommended to start with the Introduction, followed by the Product Scope and Intended Audience and Document Overview sections for an overall understanding. Developers should proceed to the System Architecture and Functional Requirements sections, while testers can focus on the Testing Requirements section. Users should refer to sections relevant to them roles, and project managers and marketing staff can benefit from the entire document for a comprehensive view.

1.3 Definitions, Acronyms and Abbreviations

1.3.1 Definitions

- **User**: An individual who interacts with the system to track expenses, income, and financial data.
- **Expense**: Any financial outflow from the user, such as purchases or bills.
- **Income**: Any financial inflow to the user, such as salary or earnings.
- **OTP (One-Time Password)**: A security mechanism used for user authentication, valid only for a short duration or a single transaction.
- **AI (Artificial Intelligence)**: Technology used for automating tasks, such as expense categorization, using machine learning or NLP.
- **NLP (Natural Language Processing)**: A subfield of AI that enables the system to understand and interpret human language inputs.

- **Kafka**: An open-source event streaming platform used for managing real-time data streams between services.
- **Kubernetes**: An open-source container orchestration platform used to automate the deployment and scaling of containerized applications.

1.3.2. Acronyms and Abbreviations

- **SRS**: Software Requirements Specification
- **CRUD**: Create, Read, Update, Delete
- **AI**: Artificial Intelligence
- **NLP**: Natural Language Processing
- **OTP**: One-Time Password
- **UI**: User Interface
- **REST**: Representational State Transfer
- **API**: Application Programming Interface
- **GCP**: Google Cloud Platform
- **DB**: Database
- **SSL**: Secure Sockets Layer
- **TLS**: Transport Layer Security
- **CI/CD**: Continuous Integration / Continuous Deployment
- **K8s**: Kubernetes (abbreviation for ease)
- **IoT**: Internet of Things

1.4 Document Convention

This document adheres to industry-standard formatting guidelines to ensure clarity, consistency, and professionalism.

Formatting Conventions:

- **Font and Spacing**:
  - The document uses Arial font with a size of 11 or 12 or 14 throughout.
  - The text is single-spaced with 1-inch margins on all sides to maintain a clean and professional layout.
- **Section Titles**:
  - All section and subsection titles are formatted in **bold** and numbered hierarchically for clarity and consistency.
- **Notes and Comments**:
  - Important notes or additional comments are presented in *italics* to distinguish them from the main content.

Typographical Conventions:

- **Special Terms and Abbreviations**:
  - Specific terms, abbreviations, and acronyms are highlighted in **bold** or capitalized to emphasize their importance and ensure consistency.
- **Lists**:
  - Bullet points are used for unordered lists, while numbered lists present ordered information.

Additional Standards:

- The document structure follows best practices for technical documentation, covering key areas such as purpose, system architecture, functional requirements, and exceptions.
- Flowcharts, diagrams, and tables include descriptive captions for better comprehension.

Revision Tracking:

- A revision history table is maintained, documenting updates with details such as Date, Version, Author, and Description.

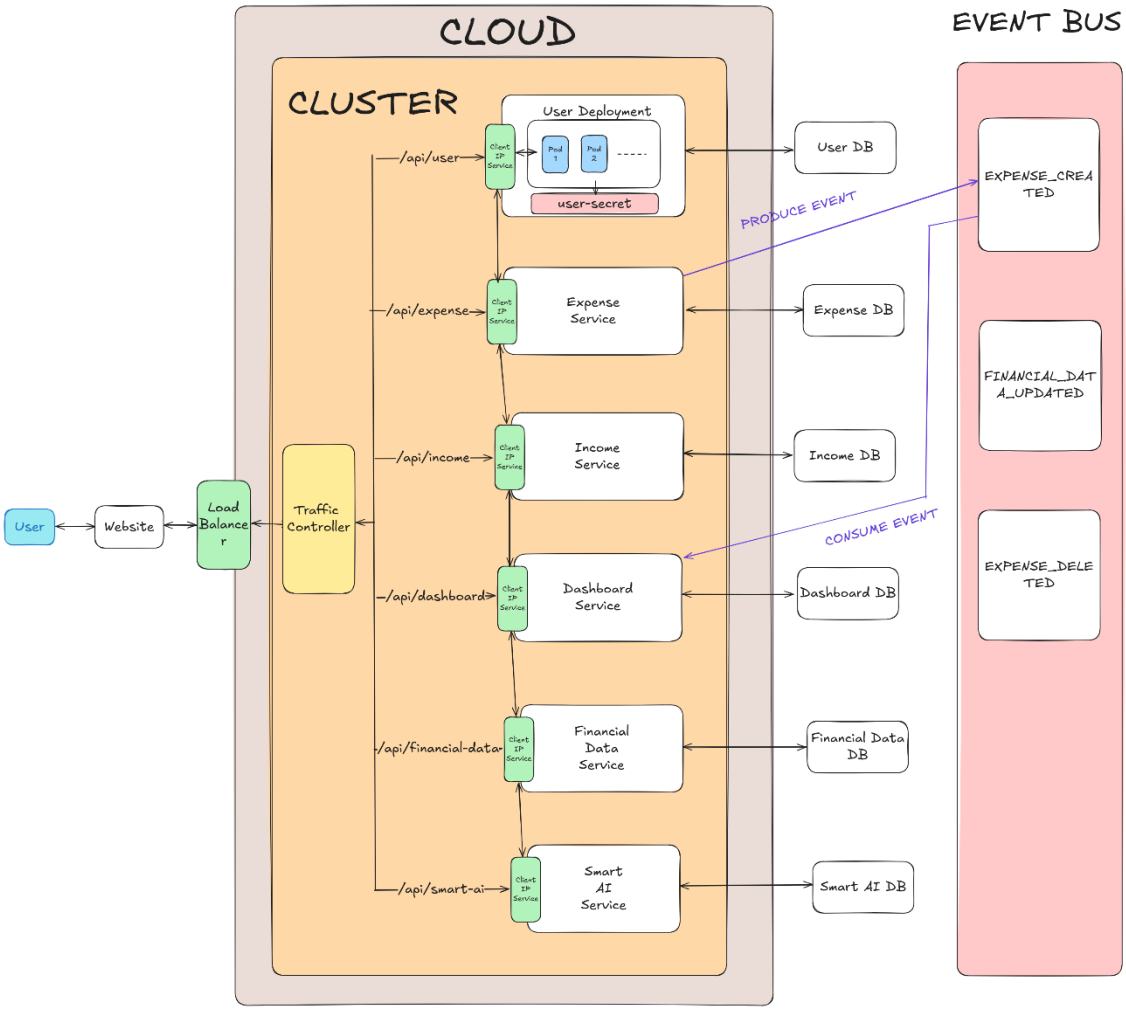
1.5 References and Acknowledgments

We'd also like to acknowledge the help of our TA, Mr. Naman Baranwal, for their valuable input in the creation of this document. We also would like to thank Prof. Indranil Saha for providing the SRS template and teaching the concepts.

2 Overall Description

2.1 Product Overview

FinXpert is a modern finance management application designed to help users track expenses and incomes, gain insights into financial habits, and manage their finances efficiently.



2.1.1 Key Features:

- Expense & Income Tracking**:** Easily add, update, and categorize financial entries with automated categorization.
- AI/NLP-Powered Chat Assistant**:** Add entries or request summaries through natural language interactions.
- Interactive Dashboard**:** Real-time analytics, trends, and summaries for comprehensive financial tracking.
- Secure Authentication**:** OTP-based login with rate-limiting and encrypted data for robust security.

- Scalable Architecture**:** Built on Kubernetes and Kafka for high performance and scalability.

2.1.2 Target Audience:

- Individuals managing personal finances.
- Small business owners needing simple financial tracking.
- Finance enthusiasts seeking actionable insights.

2.1.3 Core Benefits:

- Simplified user interface with intuitive forms and AI-driven interactions.
- Personalized insights and summaries for informed financial decisions.
- Real-time updates and analytics for proactive management.
- Mobile-friendly design and robust security for peace of mind.

2.1.4 Future Enhancements:

- Complete mobile optimization.
- Predictive AI for expense forecasting and savings advice.
- Integration with external financial tools.

FinXpert blends user-focused design with cutting-edge technology to provide an intuitive, secure and scalable platform for financial management.

2.2 Product Functionality

2.2.1. User Service

Technology: Node.js, Express, PostgreSQL

Features:

- Secure OTP authentication with rate limiting to prevent abuse.
- User profile management for handling personal and account details.
- Account lockout and recovery mechanisms for enhanced security.

2.2.2. Expense Service

Technology: Node.js, Express, MongoDB

Features:

- **CRUD operations** for managing expense entries (Create, Read, Update, Delete).
- Transactional integrity ensured using MongoDB transactions.
- Publishes events such as EXPENSE_CREATED and EXPENSE_DELETED to Kafka for communication with other services.

2.2.3. Income Service

Technology: Node.js, Express, MongoDB

Features:

- **CRUD operations** for managing income entries (Create, Read, Update, Delete).
- Transactional integrity for reliable data operations.
- Publishes events like INCOME_CREATED to Kafka for synchronization across services.

2.2.4. Financial Data Service

Technology: Node.js, Express, MongoDB

Features:

- Aggregates data from Expense and Income services for comprehensive financial insights.
- Analyzes spending habits based on categories, moods, and cognitive triggers.
- Provides data to the **Dashboard Service** and **Smart AI Assistant** for user interactions.
- Conducts detailed financial analysis to help users understand their financial patterns.

2.2.5. Dashboard Service

Technology: Node.js, Express, MongoDB

Features:

- Aggregates frequently accessed user data for instant retrieval.
- Provides real-time visualization of financial data using interactive charts and summaries.
- Consumes events from various services to ensure data accuracy and synchronization.

2.2.6. Smart AI Service

Technology: Node.js, Express, MongoDB, NLP Libraries

Features:

- Leverages **Natural Language Processing (NLP)** to understand and respond to user queries.
- Allows users to add expenses and incomes through conversational input.
- Generates personalized financial summaries tailored to individual user data.
- Responds to questions about financial data using advanced analysis and insights.

2.2.7. Frontend Service

Technology: React.js, Redux, Socket.io

Features:

- User authentication and session management for secure access.
- Provides an **interactive dashboard** with real-time updates for a seamless user experience.
- Includes a chat interface for interacting with the **Smart AI Assistant**.
- Responsive design with mobile optimization (in progress).

2.3 Design and Implementation Constraints

Hardware Limitation:

We need a computer/mobile and an internet connection for this application to build and execute.

2.3.1 Design Constraints

Microservices Architecture:
- The platform must use a fully decoupled microservices architecture to allow independent development, scaling, and deployment of services.
- Each service should adhere to the **Single Responsibility Principle**, managing only one functional area (e.g., User Service, Expense Service).

Event-Driven Communication:
- All inter-service communication must be event-driven, using Kafka to ensure scalability, resilience, and asynchronous processing.
- Services should handle eventual consistency via mechanisms like the **Distributed Saga Pattern.**

Data Management:
- MongoDB must be used for unstructured, hierarchical, or nested data (e.g., expenses, incomes, cognitive triggers).
- PostgreSQL should store relational data requiring high transactional integrity (e.g., user authentication).

Scalability & High Availability:
- All services should be horizontally scalable using Kubernetes.
- The architecture should support rolling updates and auto-scaling of services.

User Experience:
- The UI must be sleek, responsive, and intuitive, with a mobile-optimized experience.
- Features like real-time updates and conversational interfaces must be seamless and engaging for users.

Security Requirements:
- OTP-based authentication must be implemented with rate-limiting to prevent abuse.
- All communication between services and external clients must be secured via **TLS/SSL encryption**.
- Sensitive data must be stored securely using Kubernetes Secrets.

2.3.2 Implementation Constraints

Technology Stack:
- Backend: Node.js with Express.js.
- Frontend: React.js with Redux for state management and Socket.io for real-time communication.

- Event Bus: Kafka (with Zookeeper) for inter-service messaging.
- Databases: MongoDB for unstructured data, PostgreSQL for relational data.

Containerization and Orchestration:

- Each service must be containerized using Docker.
- Kubernetes must manage deployments, service discovery, load balancing, and auto-scaling.
- A NGINX Ingress Controller should route external traffic to the appropriate services.

Service Independence:

- Microservices must operate independently, with minimal downtime during updates.
- New services should be capable of replaying past events from Kafka to reconstruct state without affecting other services.

AI/NLP Requirements:

- NLP libraries must be integrated into the Smart AI Service to process natural language inputs.
- AI models should be lightweight and capable of real-time inference for financial insights.

Event Topics:

- Kafka topics must be designed with clarity (e.g., EXPENSE_CREATED, FINANCIAL_DATA_UPDATED), ensuring scalability as new events are introduced.

Infrastructure Constraints:

- The platform must be deployed on Google Cloud Platform (GCP), leveraging its managed services for scalability and resilience.
- Persistent storage solutions must be used for database data durability (e.g., Kubernetes Persistent Volume Claims).

User Data Protection:

- Compliance with data protection best practices, such as encrypting sensitive data and ensuring secure user authentication mechanisms.
- Public guest accounts should have restricted functionality to prevent misuse.

Time and Resource Constraints:

- The project was built during the Hack36 Hackathon, implying limited time for initial development.
- Contributions from open-source developers may introduce variability in skillsets and code quality.

Mobile Responsiveness:

- The mobile version of the application is a work in progress, which limits accessibility for users on smaller devices initially.

Community Collaboration:

- Contributors need clear documentation and guidelines to ensure seamless integration of new features or services.

2.4 Assumptions and Dependencies

2.4.1. Assumptions

User Behavior:

a. Users will have basic knowledge of financial tracking and can interact with the app intuitively.
b. Guest users will avoid entering sensitive information, as guest accounts are public.

Network and Infrastructure:

c. A stable and reliable internet connection will be available for users to access the app.

d. The cloud infrastructure (Google Cloud Platform) will provide consistent availability and performance.

Data Input:

e. Users will provide accurate data while adding expenses and incomes for meaningful financial analysis.
f. NLP processing assumes inputs will generally be in grammatically correct natural language.

System Load:

g. Initial usage will be manageable within the capacity of the current Kubernetes and Kafka configurations.
h. The majority of operations will not exceed the rate limits for OTP requests or other critical services.

Security Compliance:

i. Data encryption and privacy mechanisms will meet applicable legal and regulatory standards.
j. Users will follow best practices for maintaining account security, such as not sharing OTPs.

2.4.2. Dependencies

Technological Dependencies:

a. Google Cloud Platform (GCP)**:** Essential for hosting, Kubernetes orchestration, and scaling infrastructure.
b. Kafka and Zookeeper**:** Required for event-driven communication and maintaining asynchronous service operations.
c. MongoDB and PostgreSQL**:** Critical for data storage (unstructured and relational data).
d. NLP Librarie**s:** Dependency for the AI assistant to process and interpret natural language input accurately.

Third-Party Services:

e. OTP delivery relies on external email/SMS providers for authentication.
f. TLS/SSL certificates from trusted providers ensure secure communication.

Code and Frameworks:

g. Backend microservices depend on Node.js, Express.js, and their associated ecosystems.
h. The frontend requires React.js and Redux for state management and user interactions.
i. WebSocket functionality is implemented using Socket.io for real-time updates.

Deployment and Orchestration:

j. Kubernetes ensures proper container orchestration and scaling.
k. NGINX Ingress Controller routes external traffic to appropriate microservices.

Event-Driven Architecture:

l. Kafka topics must be functional and well-maintained for inter-service communication.

m. All services depend on accurate event consumption and production for smooth operations.

User-Device Compatibility:

n. The app depends on browsers supporting modern web standards for optimal performance.
o. Mobile responsiveness requires devices capable of handling complex UI interactions.

Security Measures:

p. Proper functioning of Kubernetes Secrets and encryption mechanisms for secure data handling.
q. Dependency on rate-limiting mechanisms to prevent abuse, such as brute-force OTP attempts.

3 Specific Requirements

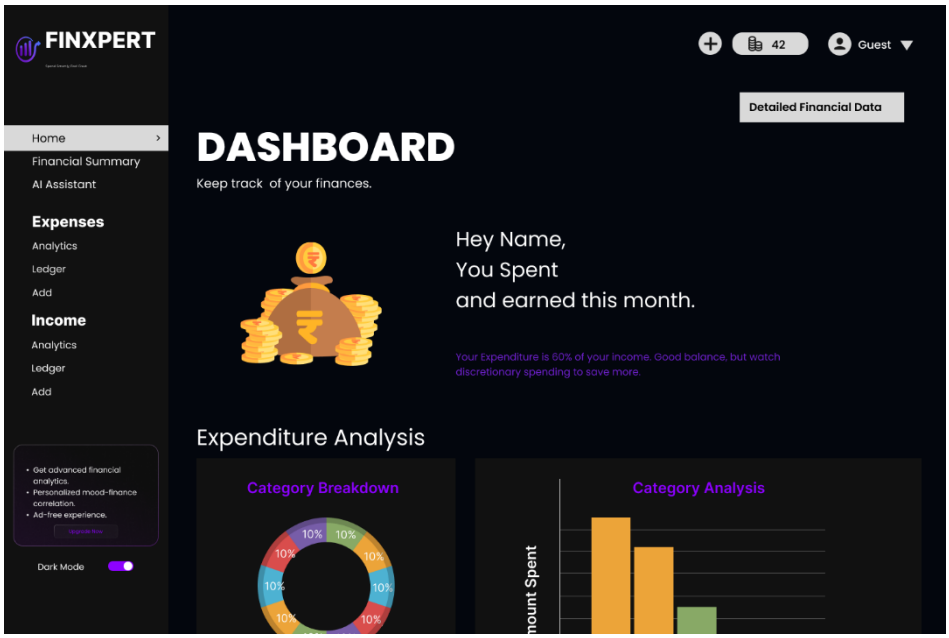
3.1 External Interface Requirements

3.1.1 User Interfaces

LOGIN PAGE

The login page of FinXpert ensures secure and seamless access to user accounts. It features a tagline, "Smart financial insights for a better you!", and options for **Guest Login** and **Contact Us** in the header. Users can enter their credentials in the username and password fields, with options to remember login details or reset passwords. New users can sign up via the provided link. A "Get on Play Store" button is available for app downloads. The page is designed with a financial-themed background, emphasizing trust and professionalism, while end-to-end encryption ensures data security.

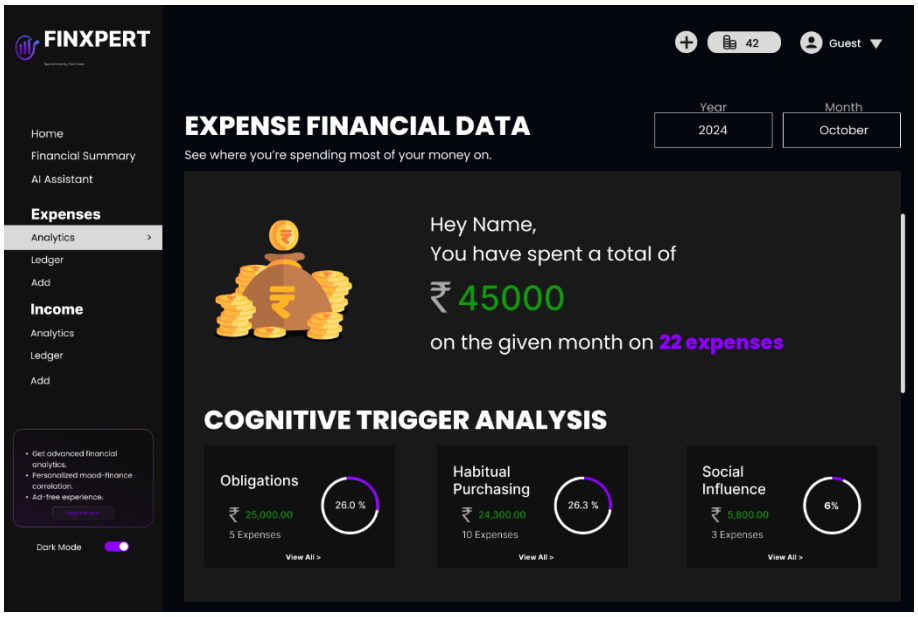
DASHBOARD



The dashboard serves as the central hub for users, providing a clear and interactive overview of their financial health. It features real-time updates on expenses, income, and savings, along with visual insights like charts and graphs to highlight spending trends and patterns. Users can easily track their progress toward financial goals and correlate mood data with financial activity for a holistic understanding of their finances.

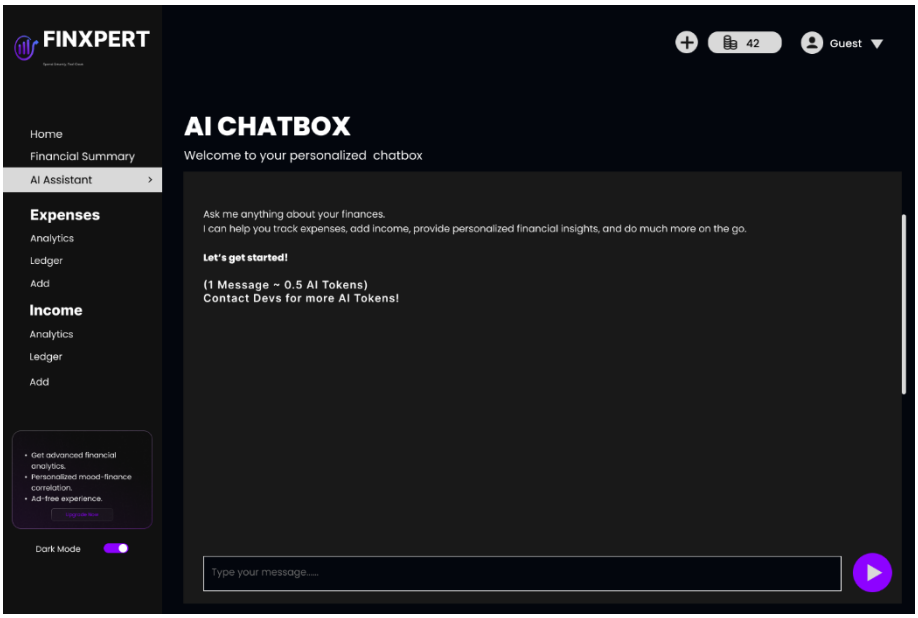
..

ANALYTICS



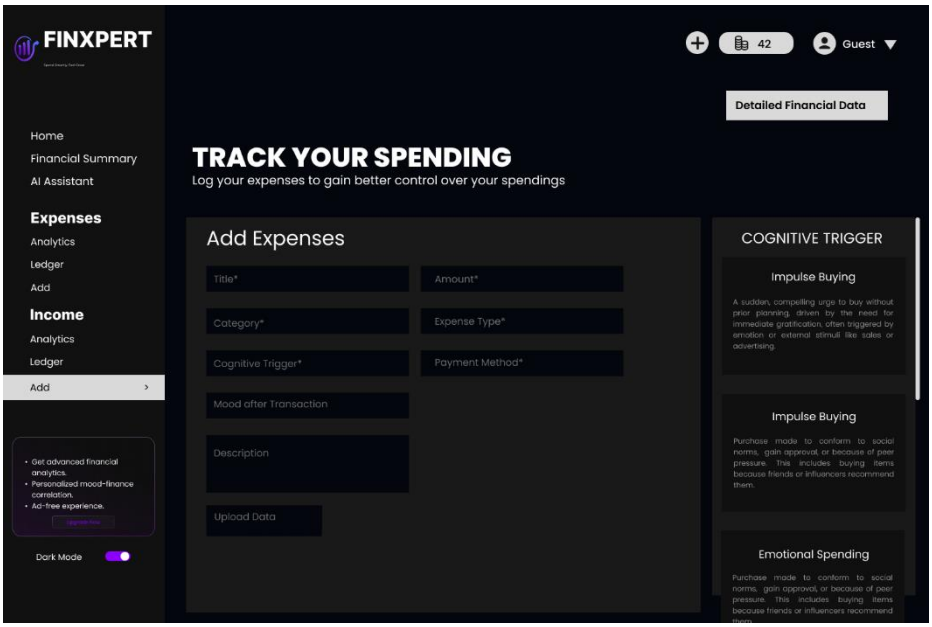
The analytics section in FinXpert provides detailed insights into expenses and income to help users understand their financial behavior better. It highlights total spending or earning for a selected month and categorizes transactions for a comprehensive analysis. The "Cognitive Trigger Analysis" offers advanced breakdowns, such as obligations, habitual purchases, and social influences, displaying the percentage contribution and corresponding amounts. Users can delve deeper by viewing detailed reports for each category. This feature empowers users to track spending patterns, manage finances effectively, and make informed decisions. A toggle for dark mode and options to upgrade for advanced analytics are also available.

AI CHATBOX



FinXpert now includes an AI-powered chatbot to assist users with their financial queries and provide personalized recommendations. The chatbot serves as a virtual assistant, offering insights into spending patterns, savings strategies, and income management. Users can ask questions, get tailored advice, and receive reminders for bill payments or investment opportunities. This feature enhances user experience by making financial management intuitive and interactive, ensuring users have access to smart, real-time assistance at their fingertips.

ADD ITEMS



The application features an "Add Item" option under the burger menu for both expenses and income sources. Users can log individual entries by providing details such as the title, mood of the transaction, cognitive triggers, and other relevant information. This functionality allows users to categorize and personalize their financial data, offering deeper insights into spending habits and income patterns. By capturing these specifics, FinXpert ensures a holistic understanding of financial behavior, empowering users to manage their finances effectively and identify trends that align with their goals.

3.1.2 Hardware Interfaces

There is no hardware interfaces involved in the project. No additional hardware required.

3.1.3 Software Interfaces

- **Database Management System (DBMS)**:

- **Connection Description**: The FinXpert system interfaces with two primary databases: **MongoDB** for unstructured data (expenses, incomes, cognitive triggers) and **PostgreSQL** for relational data (user credentials and authentication).
- **Interaction**:
    - The backend services utilize SQL (PostgreSQL) and MongoDB-specific queries for CRUD operations.
    - MongoDB handles unstructured data and uses **MongoDB Transactions** for consistency, while PostgreSQL is used for relational queries and transactional integrity.
    - Both databases ensure data consistency and durability, supporting replication for high availability and failover.
- **Web Browsers (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge)**:
- **Connection Description**: Fin Expert's user interface is built to be accessible via major web browsers.
- **Interaction**:

  - The front-end web application is optimized for compatibility across different browsers, ensuring consistent user experience on **Google Chrome**, **Mozilla Firefox**, **Safari**, and **Microsoft Edge**.
  - The system uses **responsive design** to adjust the layout and content for different screen sizes and resolutions.

- **Operating System (macOS, Windows)**:

- **Connection Description**: FinXpert interfaces with the operating system (macOS or Windows) where the server is deployed.
- **Interaction**:
  - Utilizes OS-level components for handling system calls, managing resources like CPU, memory, and storage, and managing network connections.
  - File management and log handling tasks are supported through OS-level services.
  - Ensures security and access control by interacting with the OS's built-in user permissions and access control mechanisms.

- **Web Development Framework (HTML, CSS, JavaScript)**:

- **Connection Description**: Fin Expert's front-end is developed using **HTML**, **CSS**, and **JavaScript**, supported by the **React.js** framework.
- **Interaction**:
  - **HTML** structures the content of the web pages, **CSS** handles the visual presentation (styling), and **JavaScript** provides dynamic functionality (interactivity and updates).
  - **React.js** enhances the user interface, ensuring responsiveness, interactivity, and smooth transitions, particularly for real-time updates (e.g., financial data and AI conversations).
  - **Redux** is used for state management, ensuring consistency of data across the user interface.

- **Authentication and Authorization Services (e.g., OAuth):**

- **Connection Description**: FinXpert interfaces with an authentication system that handles user authentication via **OTP (One-Time Password)** for secure login and account management.
- **Interaction**:
  - Utilizes **OTP-based authentication** for secure access control. When a user logs in, a unique OTP is generated and sent to the user's registered email or phone.
  - The system may integrate **OAuth** protocols for third-party integrations (e.g., linking with social media accounts or external financial platforms).
  - Ensures user roles and permissions are correctly enforced, providing access to different resources based on authentication status.

- **Event Streaming Platform (Apache Kafka)**:

- **Connection Description**: FinXpert utilizes **Apache Kafka** for event-driven communication between microservices.
- **Interaction**:
  - Microservices like **Expense Service**, **Income Service**, and **Smart AI Service** publish events (e.g., EXPENSE_CREATED, INCOME_CREATED) to Kafka topics.
  - Kafka ensures the reliable and efficient transmission of events between services, allowing them to remain decoupled but still exchange necessary data.
  - The event streaming setup supports **event replay**, enabling new services or components to synchronize their states with past events.

- **Cloud Infrastructure (Google Cloud Platform)**:

- **Connection Description**: FinXpert is hosted on **Google Cloud Platform (GCP)** for its cloud infrastructure, ensuring scalability and high availability.
- **Interaction**:
    - Utilizes **Google Kubernetes Engine (GKE)** for container orchestration of microservices, ensuring scalability and ease of management.
    - Leverages GCP services for networking, storage, and computing needs, such as **Google Cloud Storage** for backups and **Cloud Load Balancing** for distributing incoming traffic across multiple services.
    - **GCP Secrets Manager** securely stores sensitive information like API keys and database credentials.

- **Event Bus (Kafka, Zookeeper)**:

- **Connection Description**: The system interfaces with **Apache Kafka** as the event bus for communication between services and **Zookeeper** for coordination of Kafka brokers.
- **Interaction**:
    - Services communicate asynchronously by publishing and consuming events to/from Kafka topics, ensuring decoupled communication.
    - **Zookeeper** handles the coordination of Kafka brokers, managing partition assignments and ensuring that the system scales effectively.

3.2 Functional Requirements

F1: User Account Management

The system shall:

- Allow users to sign up using their email and complete OTP authentication.
- Enable secure OTP authentication with rate limiting to prevent abuse.
- Provide account recovery mechanisms for users who forget their credentials.
- Permit guest login for exploring the app's features, with restrictions on confidential data entry.

F2: Expense Management

The system shall:

- Allow users to perform **CRUD operations** for managing expense entries (Create, Read, Update, Delete).
- Ensure transactional integrity for expense data using MongoDB transactions.
- Publish events such as `EXPENSE_CREATED` and `EXPENSE_DELETED` to Kafka for synchronization with other services.

F3: Income Management

The system shall:

- Allow users to perform **CRUD operations** for managing income entries (Create, Read, Update, Delete).
- Ensure transactional integrity for income data.
- Publish events like `INCOME_CREATED` to Kafka for integration with other services.

F4: Financial Data Analysis

The system shall:

- Aggregate data from Expense and Income services for comprehensive financial insights.
- Analyze spending habits based on categories, moods, and cognitive triggers.
- Provide detailed financial analysis to help users understand their financial patterns.
- Deliver aggregated data to the Dashboard service and Smart AI Assistant for real-time visualization and user queries.

F5: Dashboard Functionality

The system shall:

- Aggregate frequently accessed financial data for instant retrieval.
- Provide real-time visualization of financial data through interactive charts and summaries.
- Consume events from various services to ensure up-to-date and accurate data display.

F6: Smart AI Assistant

The system shall:

- Leverage **Natural Language Processing (NLP)** to understand and respond to user queries.
- Allow users to add expenses and incomes through conversational input.
- Generate personalized financial summaries tailored to individual users.
- Respond to specific financial data queries, including trends and spending insights.

F7: Frontend Functionality

The system shall:

- Allow user authentication and session management for secure access.
- Provide an interactive dashboard with real-time updates and data visualization.
- Offer a chat interface for users to interact with the Smart AI Assistant.
- Ensure a responsive design optimized for desktop, with ongoing improvements for mobile compatibility.

F8: Security and Privacy

The system shall:

- Encrypt all user data to ensure confidentiality and security.
- Limit OTP requests to prevent spam and unauthorized access.
- Restrict the functionality of guest accounts to protect user data and maintain app integrity.

3.3 Use Case Model

Use Case 1: User Registration and Authentication

Author: Ayush Kumar Yadav

Purpose: Allow users to create an account and authenticate using secure OTP-based verification.

Requirements Traceability:

- **F1:** User Account Management – Enable secure signup and OTP authentication.
- **F8:** Security and Privacy – Encrypt user data and prevent abuse with rate limiting.

Priority: High

Preconditions:

- User must provide a valid email.
- Email service and OTP service must be operational.

Postconditions:

- The user account is created and authenticated.
- User session is initiated upon successful authentication.

Actors:

- User
- Email Service
- OTP Service

Exceptions:

- OTP delivery failures due to network issues.
- Account creation errors if the email is already registered.

Includes:

- None.

Notes/Issues:

- Consider adding CAPTCHA to prevent automated abuse.

Use Case 2: Adding an Expense via the Smart AI Assistant

Author: Rahul Mahato

Purpose: Enable users to add expense entries using conversational input through the Smart AI Assistant.

Requirements Traceability:

- **F2:** Expense Management – Support expense CRUD operations.
- **F6:** Smart AI Assistant – Process conversational input for adding expenses.

Priority: High

Preconditions:

- User must be authenticated.
- Smart AI Assistant service must be functional.

Postconditions:

- Expense data is added to the system.
- Expense category is auto-assigned if possible.

Actors:

- User
- Smart AI Assistant

Exceptions:

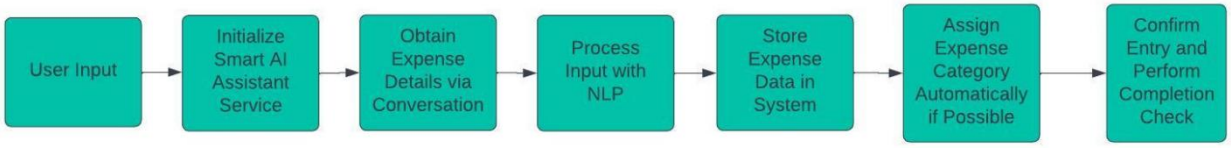
- Misunderstanding of user input by the NLP system.
- Backend service unavailability for storing the expense.

Includes:

- Use Case ID 6: Expense Categorization.

Notes/Issues:

- Continuously train the NLP model for better accuracy in understanding user queries.



Use Case 3: Generating Financial Summaries

Author: Archit Atrey

Purpose: Provide users with quick financial summaries, including total income, expenses, and savings.

Requirements Traceability:

- **F4:** Financial Data Analysis – Aggregate data for insights.
- **F5:** Dashboard Functionality – Deliver summaries to the dashboard for instant retrieval.

Priority: Medium

Preconditions:

- User must have sufficient financial data in the system.
- The system must have access to up-to-date financial records.

Postconditions:

- A summarized view of financial data is displayed on the dashboard.
- Summaries are cached for faster retrieval.

Actors:

- User
- Dashboard Service

Exceptions:

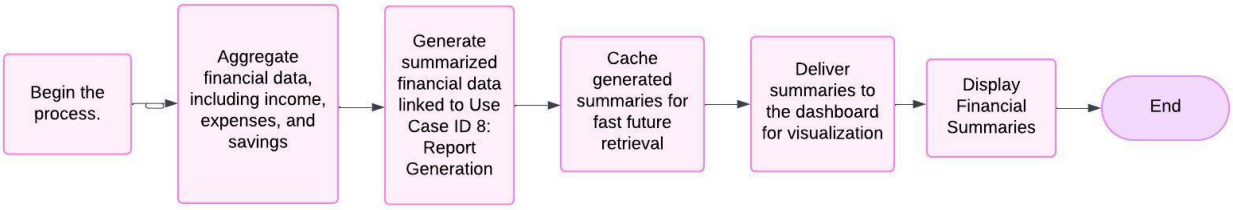
- Data aggregation errors due to inconsistent data sources.
- Dashboard visualization failures due to service outages.

Includes:

- Use Case ID 8: Report Generation.

Notes/Issues:

- Optimize aggregation for high volumes of financial data.



Use Case 4: Dashboard Visualization

Author: Sumit Kumar

Purpose: Display real-time financial data, trends, and analytics in an interactive and user-friendly dashboard.

Requirements Traceability:

- **F5:** Dashboard Functionality – Aggregate financial data for instant retrieval and visualization.
- Requirement ID 10: Dashboard must refresh automatically to reflect recent data.
- Requirement ID 11: Interactive charts and summaries should be accessible.

Priority: High

Preconditions:

- User must be logged in.
- Dashboard Service must have access to aggregated financial data.

Postconditions:

- Users can view up-to-date financial analytics.
- Dashboard interactions are logged for further analysis.

Actors:

- User
- Dashboard Service

Exceptions:

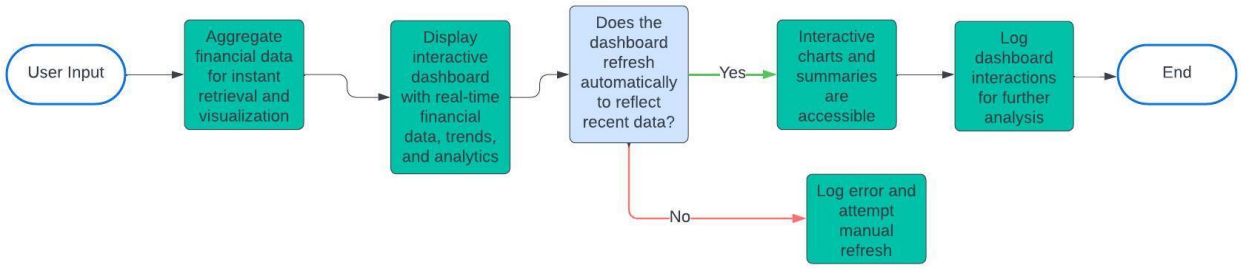
- Failure to fetch updated data from the backend.
- High latency in rendering visualizations.

Includes:

- Use Case ID 3: Generating Financial Summaries.

Notes/Issues:

- Optimize the dashboard for mobile responsiveness.
- Test scalability for large datasets to maintain performance.



Use Case 5: Secure Account Recovery Mechanism

Author: Devank Saran Prajapati

Purpose: Allow users to recover their accounts in case of forgotten credentials or account lockouts.

Requirements Traceability:

- **F1:** User Account Management – Enable secure recovery mechanisms.
- **F8:** Security and Privacy – Encrypt data and log recovery attempts for audits.
- Requirement ID 12: Users must receive recovery options via email.
- Requirement ID 13: Secure authentication must validate user identity during recovery.

Priority: Medium

Preconditions:

- User must have a registered email address.
- The OTP generation service must be functional.

Postconditions:

- The user regains access to their account.
- All recovery attempts are logged for security audits.

Actors:

- User
- OTP generation service
- Email service provider

Exceptions:

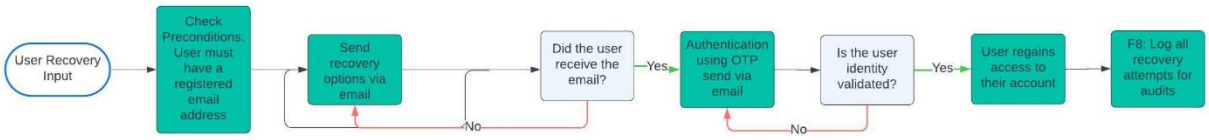
- Incorrect answers to security questions (if implemented).
- Delivery issues with recovery emails.

Includes:

- Use Case ID 1: User Registration and Authentication.

Notes/Issues:

- Evaluate the need for multi-factor authentication during recovery.
- Consider rate limiting recovery attempts to prevent abuse.



Use Case 6: Expense Categorization

Author: Karthik S. Pillai

Purpose: Automatically categorize expenses based on user input and AI-generated insights.

Requirements Traceability:

- **F2:** Expense Management – Allow CRUD operations for expense entries.
- **F4:** Financial Data Analysis – Analyze spending habits and provide insights for better categorization.

Priority: High

Preconditions:

- Expense data must be added to the system.
- AI categorization service must be functional.

Postconditions:

- The expense is assigned an appropriate category.
- Category modifications, if any, are updated in real time.

Actors:

- User
- AI Categorization Service

Exceptions:

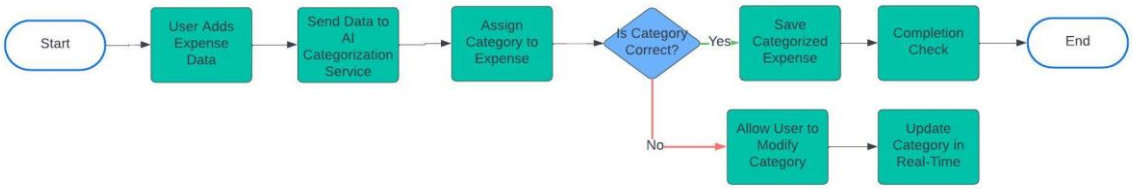
- Misclassification of expenses due to insufficient data.
- Category modification failure due to service unavailability.

Includes:

- Use Case ID 2: Adding an Expense via the Smart AI Assistant.

Notes/Issues:

- Implement a feedback loop to improve categorization accuracy over time.



Use Case 7: Budget Tracking and Alerts

Author: Dwij Om Oshoin

Purpose: Enable users to set budgets and receive alerts when nearing or exceeding budget limits.

Requirements Traceability:

- **F4:** Financial Data Analysis – Aggregate and analyze spending habits.
- **F5:** Dashboard Functionality – Provide real-time visualizations for budget tracking.

Priority: Medium

Preconditions:

- Users must define a budget.
- The system must have access to real-time expense data.

Postconditions:

- Users are notified of budget status via in-app alerts or email.
- Budget data is logged for analysis and reporting.

Actors:

- User
- Notification Service

Exceptions:

- Delay in real-time budget alerts due to system lag.
- Incorrect budget calculation caused by unprocessed expense data.

Includes:

- Use Case ID 6: Expense Categorization.

Notes/Issues:

- Add a feature for predictive alerts based on spending trends.

Use Case 8: Report Generation

Author: Madhav Khetan

Purpose: Generate detailed financial reports for a selected period, including income, expenses, and savings.

Requirements Traceability:

- **F4:** Financial Data Analysis – Aggregate expense and income data for insights.
- **F5:** Dashboard Functionality – Deliver aggregated data to the dashboard for visualization.

Priority: Medium

Preconditions:

- User must select a date range for the report.
- Report generation service must have access to financial data.

Postconditions:

- The user receives a downloadable report.
- The report is stored for future reference in the user's account.

Actors:

- User
- Report on Generation Service

Exceptions:

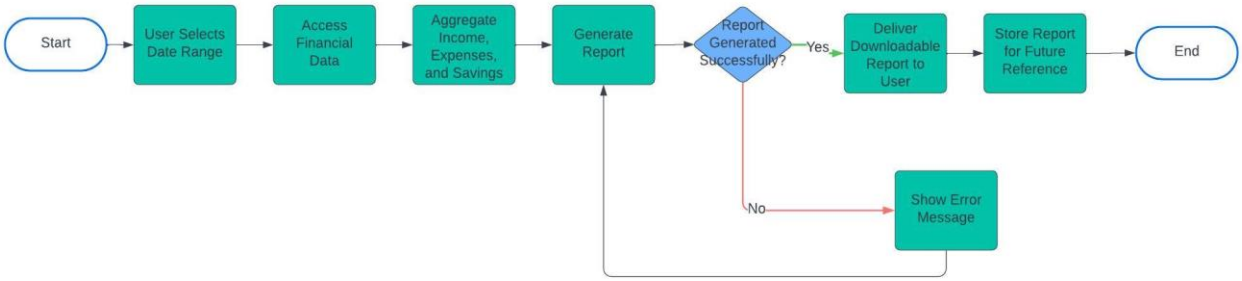
- Incomplete report generation due to backend service failure.
- Export functionality errors for specific formats.

Includes:

- Use Case ID 3: Generating Financial Summaries.

Notes/Issues:

- Implement scheduled report generation for automated delivery.



Use Case 9: Sharing Financial Insights

Author: Piyush Meena

Purpose: Allow users to share specific financial insights or reports with others via email or social media.

Requirements Traceability:

- **F5:** Dashboard Functionality – Aggregate and share financial data.
- **F8:** Security and Privacy – Ensure secure sharing and restricted access.

Priority: Low

Preconditions:

- The user must specify recipients and permissions.
- Sharing service must validate recipient details.

Postconditions:

- Selected data is securely shared with recipients.
- Recipients receive notifications with access instructions.

Actors:

- User
- Sharing Service
- Notification Service

Exceptions:

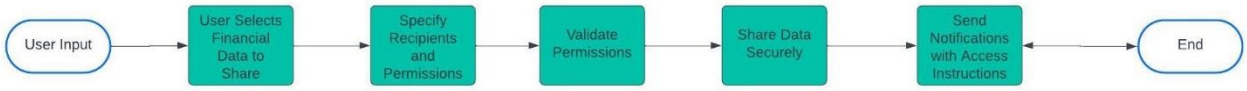
- Sharing failure due to incorrect recipient details.
- Unauthorized access due to improper permission settings.

Includes:

- Use Case ID 8: Report Generation.

Notes/Issues:

- Consider introducing watermarking for sensitive shared reports.



Use Case 10: Income Management

Author: Aniket Kumar Chaudhary

Purpose: Allow users to add and manage income sources and track income history.

Requirements Traceability:

- **F3:** Income Management – Allow CRUD operations for income entries.
- **F4:** Financial Data Analysis – Aggregate income data for financial insights.

Priority: High

Preconditions:

- User must be logged in.
- The system must have an operational income tracking module.

Postconditions:

- Income data is updated and displayed accurately in the dashboard.
- Changes are logged for future reference.

Actors:

- User
- Income Management Service

Exceptions:

- Failure to save new or updated income data due to backend issues.
- Discrepancies between income and expense data synchronization.

Includes:

- Use Case ID 3: Generating Financial Summaries.

Notes/Issues:

- Include integration with external payroll systems for automated updates.

4 Other Non-functional Requirements

4.1. Performance Requirements

4.1.1. Response Time

- **Requirement:** The system should respond to user requests within **2 seconds** for **95% of transactions**.
- **Rationale:** Ensures a seamless and efficient user experience, particularly critical for interactions with the Smart AI Assistant and dashboard visualization.

4.1.2. Concurrent User Handling

- **Requirement:** The system must support at least **500 concurrent users** without significant performance degradation.
- **Rationale:** Guarantees scalability during peak periods, such as end-of-month financial reviews or promotional events.

4.1.3. Data Retrieval Time

Requirement: Retrieval of financial data, including analytics and summaries, should not exceed 3 sec.

Rationale: Quick data access ensures timely insights, maintaining user satisfaction and engagement.

4.1.4. Transaction Processing

- **Expense/Income Updates:**
  - **Requirement:** Each expense or income entry, whether created, updated, or deleted, should be processed within **2 seconds**.
  - **Rationale:** Allows users to manage financial data efficiently without noticeable delays.
- **Event Processing via Kafka:**
  - **Requirement:** Events published to Kafka should be consumed and processed within **500 milli sec.**
  - **Rationale:** Maintains system consistency and ensures real-time updates across services.

4.1.5. Authentication Time

- **Requirement:** User authentication (OTP-based) must complete within **1 second**.
- **Rationale:** Enhances user experience by providing quick access to the platform, especially during initial login or re-authentication.

4.1.6. System Uptime

- **Requirement:** Minimum uptime of **99.5%**.

- **Rationale:** Ensures reliable service availability, particularly for time-sensitive features like financial summaries and transaction logging.

4.1.7. Dashboard Update Time

- **Requirement:** Dashboard updates (including real-time visualizations) should reflect changes in the data within **2 seconds**.
- **Rationale:** Provides users with an accurate and up-to-date view of their financial data, improving decision-making.

4.1.8. Data Storage Capacity

- **Requirement:** The database must accommodate a growth rate of **20% per year** over the next three years.
- **Rationale:** Prepares for future scalability, ensuring smooth operations without the need for frequent storage upgrades.

4.1.9. Booking Confirmation Time

- **Requirement:** For any financial action (e.g., expense addition or modification), the system should provide confirmation to the user within **5 seconds**.
- **Rationale:** Immediate feedback ensures user confidence in the accuracy and reliability of the system.

4.1.10. AI Assistant Query Response

- **Requirement:** Responses to queries from the Smart AI Assistant should be generated within **1 second** for simple queries and **3 seconds** for complex ones.
- **Rationale:** Ensures a conversational flow, minimizing user frustration and maintaining engagement.

4.1.11. Event Replay

- **Requirement:** New services should replay past events from Kafka and initialize their state within **1 minute** for up to **10,000 events**.
- **Rationale:** Allows quick bootstrapping of services without disrupting ongoing operations.

4.2. Safety and Security Requirements

4.2.1. User Authentication

- **Requirement:** All users must go through a secure authentication process, including OTP-based verification, to access the system.
- **Rationale:** Protects user accounts from unauthorized access, ensuring confidentiality and integrity of user data.

4.2.2. Password Policies

- **Requirement:** Users must create strong passwords with at least 8 characters, including uppercase, lowercase, numbers, and special characters.
- **Rationale:** Enhances account security and reduces the risk of brute-force attacks.

4.2.3. User Verification Process

- **Requirement:** During user registration, a one-time verification code will be sent to the registered email address. Users must use this code to validate their identity.
- **Rationale:** Confirms the legitimacy of new users and prevents unauthorized registrations.

4.2.4. Session Management

- **Requirement:** User sessions must expire after **15 minutes** of inactivity, requiring re-authentication.
- **Rationale:** Minimizes risks from unattended sessions, protecting user data.

4.2.5. Data Encryption

- **Requirement:** All sensitive data, including credentials and financial details, must be transmitted using HTTPS with TLS encryption.
- **Rationale:** Prevents eavesdropping and ensures the integrity of data during transmission.

4.2.6. Role-Based Access Control (RBAC)

- **Requirement:** Implement RBAC to restrict access based on user roles. For example:
  - Users can only manage their own financial data.
  - Administrators can configure system settings and monitor activities.
- **Rationale:** Ensures that users can only perform actions within their scope, preventing unauthorized data access or modifications.

4.2.7. Audit Trail

- **Requirement:** Maintain a detailed log of user activities, including login/logout events, financial transactions, and data modifications.
- **Rationale:** Facilitates monitoring, detecting suspicious behavior, and investigating security incidents.

4.2.8. Data Backup and Recovery

- **Requirement:** Perform regular automated backups (daily) of all critical data and maintain a recovery plan to restore operations within **24 hours** of a failure.
- **Rationale:** Safeguards against data loss due to accidental deletion, system failures, or security breaches.

4.2.9. Error Handling

- **Requirement:** Display user-friendly error messages that do not reveal sensitive system information.
- **Rationale:** Prevents attackers from gaining insights into system architecture or vulnerabilities.

4.2.10. Security Training

- **Requirement:** Developers, administrators, and other responsible personnel must undergo regular security training to stay updated on potential threats and best practices.
- **Rationale:** Promotes a security-aware culture, reducing risks from human error.

4.2.11. Incident Response Plan

- **Requirement:** Develop and document an incident response plan outlining steps to detect, contain, and recover from security incidents.
- **Rationale:** Enables a swift and organized response, minimizing the impact of breaches or vulnerabilities.

4.2.12. Database Security

- **Requirement:** Databases must be secured with access controls, encryption for sensitive fields (e.g., passwords, financial records), and regular security audits.
- **Rationale:** Protects stored data from unauthorized access or tampering.

4.2.13. Rate Limiting

- **Requirement:** Implement rate limiting on sensitive endpoints, such as OTP requests and login attempts, to prevent brute-force and denial-of-service attacks.
- **Rationale:** Reduces the risk of abuse while maintaining service availability.

4.2.14. Security Updates

- **Requirement:** Apply patches and updates to all software components, including operating systems, libraries, and frameworks, within **7 days** of release.
- **Rationale:** Prevents exploitation of known vulnerabilities.

4.2.15. Secure Configuration

- **Requirement:**

Use secure configurations for all system components, such as disabling unused ports and enforcing least-privilege access.

- **Rationale:**

Minimizes the system's attack surface and potential vulnerabilities.

4.3 Software Quality Attributes

4.3.1 Reliability

- **Transaction Reliability:** Ensure a 99.9% success rate for processing user actions like bookings.

- **Error Handling:** Log errors, provide clear user messages, and recover automatically while preserving data integrity.
- **Event Processing:** Use Kafka with at-least-once delivery and replay capabilities for inter-service reliability.

4.3.2 Usability

- **User Interface:** Design for simplicity with a SUS score of 80+.
- **Task Efficiency:** Complete common tasks in 3 steps or less.
- **Accessibility:** Comply with WCAG 2.1 standards for inclusivity.
- **Mobile Responsiveness:** Ensure an optimized experience for mobile users.

4.3.3 Performance

- **Response Time:** 95% of actions should respond within 2 seconds.
- **Scalability:** Support 500+ concurrent users without degradation.
- **Real-Time Updates:** Reflect financial updates across the platform within 3 seconds.

4.3.4 Security

- **Data Protection:** Encrypt sensitive data in transit and at rest.
- **Session Management:** Terminate inactive sessions after 15 minutes.
- **Access Control:** Implement RBAC for sensitive operations.

4.3.5 Maintainability

- **Modular Code:** Enable independent updates for microservices.
- **Version Control:** Maintain detailed logs and backward compatibility.
- **Documentation:** Provide clear developer and admin guides.

4.3.6 Availability

- **Uptime:** Ensure 99.5% monthly availability.
- **Fault Tolerance:** Handle individual service failures without disrupting operations.
- **Disaster Recovery:** Restore full functionality within 1 hour of a critical failure.

5 Other Requirements

Code Reusability: Requirement: The development team shall follow modular coding practices to facilitate code reuse across different system modules. Rationale: Promoting code reusability enhances maintainability, reduces development time, and ensures consistency in the system's implementation.

Documentation Reusability: Requirement: The documentation should be structured in a reusable format to support future system updates and expansions. Rationale: Reusable documentation formats contribute to ease of understanding, knowledge transfer, and efficient updates to the system.

Appendix A – Data Dictionary

Item	Type	Description	Possible Status	Operations
User id	Identifier	Unique identifier for each user (Student, Admin)	N/A	Used for login, user authentication
Username	String	User's chosen login name	N/A	Must be unique. Validated during user registration
Password	String	User Password for system authentication	N/A	Must meet security policies (length, special characters).
OTP (One Time Password)	String	One time password for two-factor authentication.	N/A	Sent to the user for validation during login. Valid for 10 minutes.
Expense Amount	Numeric	The amount for an individual expense	Positive Decimal	Validated for non-negative values during entry.
Income Amount	Numeric	The amount for an individual income entity	Positive Decimal	Validated for non-negative values during entry
Category	String	Category for classifying expenses / incomes (e.g.: Food, Transport)	Predefined Categories	Auto-categorization upon entry, user-defined categories for custom options.
Mood	String	User's mood associated with the expense or income (e.g., Happy).	Predefined Moods (Happy, Sad, Neutral, etc.)	Optional for tagging, linked with analytics.
Cognitive Trigger	String	Cognitive trigger that led to the expense (e.g., Stress)	Predefined Triggers (Stress, Reward, etc.)	Optional for tagging, linked with behavioral analysis.
Transaction Timestamp	Date Time	The timestamp of when the transaction was logged.	Current Date Time	Automatically captured during transaction entry.
Transaction ID	Identifier	Unique ID for each financial transaction.	N/A	Generated upon transaction creation.
User Role	String	Role assigned to the user (Student, Admin, Staff).	Student, Admin, Staff	Enforces role-based access control.
User Session Status	Boolean	Status of the user's current	Active, Inactive	Determines if the session is active or

		session (Active/Inactive)		has expired due to inactivity.
Session Timeout	Integer	Time in minutes before the session expires due to inactivity.	N/A	Set to 15 minutes as per security requirements.
Data Encryption	Boolean	Indicates if the data is encrypted during transmission.	True, False	Required for all sensitive data. Should be True (enabled) during all transactions.
Backup Status	String	The current status of the data backup operation.	In Progress, Completed, Failed	Must run periodic backups as part of the disaster recovery plan.
Audit Log Entry	String	A log of user actions and system events for security monitoring.	N/A	Captures all critical actions (e.g., logins, modifications).
API Rate Limit	Integer	The number of API requests a user can make in a given time frame.	N/A	Set to 100 requests per minute per user to prevent abuse.
Financial Summary	String	A summary of the user's financial status.	N/A	Generated automatically at the end of each month or on request.
Dashboard Data		Data displayed in the financial dashboard (expenses, incomes)	N/A	Aggregated data from all financial transactions, presented visually.

Appendix B - Group Log

Meeting with the TA was approximately 3-4 hours understanding the various aspect of the document.

12 January 2024	Brainstormed different project ideas and finalised a web application for Intelligent Financial Management Platform.
21 January 2024	Started Working on a draft of Software Requirement Specification document.
23 January 2024	Completed SRS document
24 January 2024	Discussed with TA, made necessary changes for final submission.